Week 4

#### Homework for Lesson 4

Homework is your chance to put what you've learned in this lesson into practice. This homework is not "graded" and you are encouraged to write additional code beyond what is asked.

#### Note:

- Ensure you completed the setup instructions provided on the course page, before attempting the homework.
- The solutions to the homework are NOT provided. We encourage you to try it out and discuss in the course forum for further learning.
- The homework is NOT mandatory to get the course completion award.
- Post your questions, comments, or suggestions (if any) in the course forum @ https://community.oracle.com/community/technology\_network\_community/moocs /sql-fundamentals-2018/week-4

#### Watch out for:



- Reference video that discussed the corresponding concept in this MOOC.



- Expected output.

#### Assignment 1

The HR department wants you to create SQL statements to insert, update, and delete employee data. As a prototype, you use the MY EMPLOYEE table before giving the statements to the HR department.

**NOTE:** Before starting this task, execute the cleanup 4a.sql script from the labs/cleanup scripts folder.

1. Run the lab 04 01.sql script from labs/lab scripts folder to create the MY EMPLOYEE table. Describe the structure of the MY EMPLOYEE table to identify the column names.

DESCRIBE my	/_employed	e
Name	Null	Type
ID LAST_NAME FIRST_NAME USERID SALARY	NOT NULL	NUMBER(4) VARCHAR2(25) VARCHAR2(25) VARCHAR2(8) NUMBER(9,2)



2. Create an INSERT statement to add the *first row* of data to the MY\_EMPLOYEE table from the following sample data. Do not list the columns in the INSERT clause. Do not enter all rows yet.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	cnewman	750
5	Ropeburn	Audrey	aropebur	1550

- 3. Populate the MY\_EMPLOYEE table with the second row of the sample data from the preceding list. This time, list the columns explicitly in the INSERT clause.
- 4. Confirm your addition to the table.





5. Write an INSERT statement in a dynamic reusable script file to load the next two rows of sample data into the MY\_EMPLOYEE table. The script should prompt for all the columns (ID, LAST\_NAME, FIRST\_NAME, USERID, and SALARY). Save this script to hw4\_task1\_05.sq1 file.

- 6. Populate the table with the next two rows of the sample data listed in step 3 by running the INSERT statement in the script that you created.
- 7. Confirm your additions to the table and make the data additions permanent.

	2 ID	2 LAST_NAME	FIRST_NAME	2 USERID	2 SALARY
1	1	Pate1	Ralph	rpatel	895
2	2	Dancs	Betty	bdancs	860
3	3	Biri	Ben	bbiri	1100
4	4	Newman	Chad	cnewman	750



- 8. Change the last name of employee 3 to Drexler.
- 9. Change the salary to \$1,000 for all employees who have a salary less than \$900.
- 10. Verify your changes to the table.





11. Delete Betty Dancs from the MY\_EMPLOYEE table. Confirm your changes to the table.

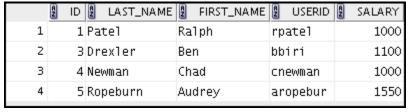




- 12. Commit all pending changes.
- 13. Populate the table with the last row of the sample data listed in step 3 by using the statements in the script that you created in step 6. Run the statements in the script.

Note: Perform the steps (15-21) in one session only.

14. Confirm your addition to the table.





- 15. Mark an intermediate point in the processing of the transaction.
- 16. Delete all the rows from the MY\_EMPLOYEE table.

- 17. Confirm that the table is empty.
- 18. Discard the most recent DELETE operation without discarding the earlier INSERT operation.
- 19. Confirm that the new row is still intact.

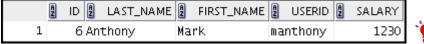




- 20. Make the data addition permanent.
- 21. Modify the hw4\_task1\_05.sql script such that the <code>USERID</code> is generated automatically by concatenating the first letter of the first name and the first seven characters of the last name. The generated <code>USERID</code> must be in lowercase. Therefore, the script should not prompt for the <code>USERID</code>. Save this script to a file <code>named</code> <code>hw4\_task1\_21.sql</code>.
- 22. Run the hw4 task1 21.sql script to insert the following record:

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
6	Anthony	Mark	manthony	1230

23. Confirm that the new row was added with the correct USERID.





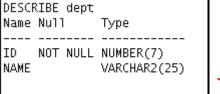
See 4-2: Inserting Data and 4-3: Modifying and Deleting Data for reference.

### Assignment 2

**NOTE:** Before starting this task, execute the cleanup\_4b.sql script from the labs/cleanup scripts folder.

1. Create the DEPT table based on the following table instance chart. Save the statement in the  $hw4\_task2\_01.sql$  script, and then execute the statement in the script to create the table. Confirm that the table is created.

Column Name	ID		NAME
Key Type	Primary	key	
Nulls/Unique			
FK Table			
FK Column	(		
Data type	NUMBER		VARCHAR2
Length	7	<u></u>	25





2. Create the EMP table based on the following table instance chart. Save the statement in the hw4\_task2\_02.sql script, and then execute the statement in the script to create the table. Confirm that the table is created.

Column Name	ID	LAST_NAME	FIRST_NAME	DEPT_ID
Key Type				
Nulls/Unique		<del>-</del> 2		
FK Table		9		DEPT
FK Column		(0)		ID
Data type	NUMBER	VARCHAR2	VARCHAR2	NUMBER
Length	7	25	25	7

Name	Nu11	Type
ID LAST_NAME FIRST_NAME DEPT_ID		NUMBER(7) VARCHAR2(25) VARCHAR2(25) NUMBER(7)



3. Modify the EMP table. Add a COMMISSION column of the NUMBER data type, with precision 2 and scale 2. Confirm your modification.

Table EMP a	alter	ed.
Name	Nu11	Туре
ID		NUMBER(7)
LAST_NAME		VARCHAR2(25)
FIRST_NAME		VARCHAR2(25)
DEPT_ID		NUMBER(7)
COMMISSION		NUMBER(2,2)

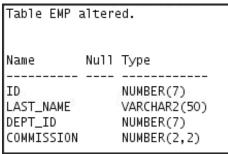


4. Modify the EMP table to allow for longer employee last names. Confirm your modification.

Table EMP altered.		
Name	Nu11	Туре
ID LAST_NAME FIRST_NAME DEPT_ID COMMISSION		NUMBER(7) VARCHAR2(50) VARCHAR2(25) NUMBER(7) NUMBER(2,2)

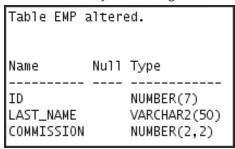


5. Drop the FIRST\_NAME column from the EMP table. Confirm your modification by checking the description of the table.





6. In the EMP table, mark the DEPT\_ID column as UNUSED. Confirm your modification by checking the description of the table.





- 7. Drop all of the UNUSED columns from the EMP table.
- 8. Create the EMPLOYEES2 table based on the structure of the EMPLOYEES table. Include only the EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, SALARY and DEPARTMENT\_ID columns. Name the columns in your new table ID, FIRST\_NAME, LAST\_NAME, SALARY and DEPT\_ID respectively. Confirm that the table is created.

Name	Nu11	Type
ID FIRST_NAME LAST_NAME SALARY DEPT_ID	NOT NULL	NUMBER(6) VARCHAR2(20) VARCHAR2(25) NUMBER(8,2) NUMBER(4)

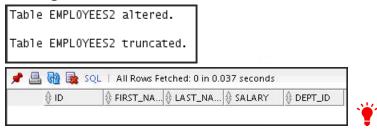


9. Alter the status of the EMPLOYEES2 table to READ ONLY.

10. Try to delete all rows in the EMPLOYEES2 table. Observe what happens.

```
Error starting at line: 7 in command -
truncate table employees2
Error report -
SQL Error: ORA-12081: update operation not allowed on table "ORA1"."EMPLOYEES2"
12081. 00000 - "update operation not allowed on table \"%s\".\"%s\""
*Cause: An attempt was made to update a read-only materialized view.
*Action: No action required. Only Oracle is allowed to update a
read-only materialized view.
```

11. Revert the EMPLOYEES2 table to the READ WRITE status. Now try to truncate the rows again.



12. Drop the EMP, DEPT, and EMPLOYEES2 table.

See <u>4-4: Creating Tables</u>, <u>4-5: Defining Constraints</u>, <u>4-6: Modifying Data Structures</u> and 4-7: Handling Transactions for reference.

#### **Assignment 3**

1. Grant another user query privilege on your table. Then, verify whether that user can use the privilege.

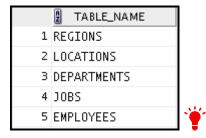
**Note:** For this exercise, open another SQL Developer session and connect as a different user. For example, if you are currently using oral, open another SQL Developer session and connect as orall.

- Grant another user (for example, ora22) privilege to view records in your REGIONS table. Include an option for this user to further grant this privilege to other users.
- Have the user query your REGIONS table.

- Have the user pass on the query privilege to a third user, ora23.
- Take back the privilege from the user who performs step b.
- 2. Grant another user query and data manipulation privileges on your COUNTRIES table. Make sure that the user cannot pass on these privileges to other users.
- 3. Take back the privileges on the COUNTRIES table granted to another user.
- 4. Grant another user access to your DEPARTMENTS table. Have the user grant you query access to his or her DEPARTMENTS table.
- 5. Query all the rows in your DEPARTMENTS table.
- 6. Add a new row to your DEPARTMENTS table. ora1 should add Education as department number 500. ora22 should add Human Resources as department number 510. Query ora22's table from ora1 and vice versa.
- 7. Revoke the SELECT privilege from ora22 and vice versa.
- 8. Remove the row that you inserted into the DEPARTMENTS table in step 6 and save the changes.
- See 4-8: Controlling User Access for reference.

### Assignment 4

1. Query the USER\_TABLES data dictionary view to see information about the tables that you own.



. . .

2. Query the ALL\_TABLES data dictionary view to see information about all the tables that you can access. Exclude the tables that you own.

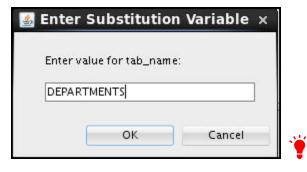
	TABLE_NAME	2 OWNER
1	DUAL	SYS
2	SYSTEM_PRIVILEGE_MAP	SYS
3	TABLE_PRIVILEGE_MAP	SYS
4	USER_PRIVILEGE_MAP	SYS
5	STMT_AUDIT_OPTION_MAP	SYS
6	AUDIT_ACTIONS	SYS
7	WRR\$_REPLAY_CALL_FILTER	SYS
8	HS_BULKLOAD_VIEW_OBJ	SYS
9	HS\$_PARALLEL_METADATA	SYS
10	HS_PARTITION_COL_NAME	SYS
11	HS_PARTITION_COL_TYPE	SYS

. . .

98	SDO_TOPO_DATA\$	MDSYS
99	SDO_GR_MOSAIC_O	MDSYS
100	SDO_GR_MOSAIC_1	MDSYS
101	SDO_GR_MOSAIC_2	MDSYS
102	SDO_GR_MOSAIC_3	MDSYS
103	SDO_GR_PARALLEL	MDSYS
104	SDO_GR_RDT_1	MDSYS
105	SDO_WFS_LOCAL_TXNS	MDSYS



3. For a specified table, create a script that reports the column names, data types, and data types' lengths, as well as whether nulls are allowed. Prompt the user to enter the table name. Give appropriate aliases to the DATA\_PRECISION and DATA\_SCALE columns. Save this script in a file named hw4\_task4\_03.sql. For example, if the user enters DEPARTMENTS, the following output results:



			∯ PRECISION	<b>♦</b> SCALE	∯ NULLABLE
1 MANAGER_ID	NUMBER	22	6	0	Υ
2 LOCATION_ID	NUMBER	22	4	0	Υ
3 DEPARTMENT_ID	NUMBER	22	4	0	N
4 DEPARTMENT_NAME	VARCHAR2	30	(nu11)	(null)	N

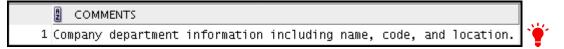
4. Create a script that reports the column name, constraint name, constraint type, search condition, and status for a specified table. You must join the USER\_CONSTRAINTS and USER\_CONS\_COLUMNS tables to obtain all this information. Prompt the user to enter the table name. Save the script in a file named hw4\_task4\_04.sql.

For example, if the user enters DEPARTMENTS, the following output results:



∯ COLU	MN_NAME &C	ONSTRAINT_NAME	⊕ CONSTRAINT_TYPE	SEARCH_CONDITION	<b>⊕</b> STATUS
1 DEPART	MENT_NAME DEP	T_NAME_NN	C	"DEPARTMENT_NAME" IS NOT NULL	ENABLED
2 LOCATIO	N_ID DEP	T_LOC_FK	R	(null)	ENABLED
3 MANAGER	⊂ID DEP	T_MGR_FK	R	(null)	ENABLED
4 DEPART	MENT_ID DEP	T_ID_PK	P	(null)	ENABLED

5. Add a comment to the DEPARTMENTS table. Then query the USER TAB COMMENTS view to verify that the comment is present.



6. Run the lab 04 06 tab.sql script in the labs/lab scripts folder as a prerequisite for exercises 6 through 9.

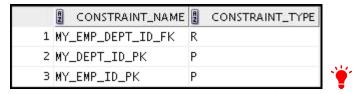
Alternatively, open the script file to copy the code and paste it into your SQL Worksheet.

Then execute the script. This script:

- Drops the existing DEPT2 and EMP2 tables
- Creates the DEPT2 and EMP2 tables
- 7. Confirm that both the DEPT2 and EMP2 tables are stored in the data dictionary.



8. Confirm that the constraints were added, by querying the <code>USER\_CONSTRAINTS</code> view. Note the types and names of the constraints.



9. Display the object names and types from the USER\_OBJECTS data dictionary view for the EMP2 and DEPT2 tables.





See 4-9: Introducing Data Dictionary Views for reference.

Congratulations you have successfully completed homework for Week 4 of SQL Fundamentals.